C.2

# BRANCH-AND-BOUND STRATEGIES FOR DYNAMIC PROGRAMMING

Thomas L. Morin* and

Roy E. Marsten**

WP 750-74 REVISED May 1975

# BRANCH-AND-BOUND STRATEGIES FOR DYNAMIC PROGRAMMING

Thomas L. Morin* and

Roy E. Marsten**

WP 750-74 REVISED May 1975

*Northwestern University, Evanston, Illinois

**Massachusetts Institute of Technology
Alfred P. Sloan School of Management, Cambridge, Massachusetts

ABSTRACT

This paper shows how branch-and-bound methods can be used to reduce storage and, possibly, computational requirements in discrete dynamic programs. Relaxations and fathoming criteria are used to identify and to eliminate states whose corresponding subpolicies could not lead to optimal policies. The general dynamic programming/branch-and-bound approach is applied to the traveling-salesman problem and the nonlinear knapsack problem. The reported computational experience demonstrates the dramatic savings in both computer storage and computational requirements which were effected utilizing the hybrid approach.

Consider the following functional equation of dynamic programming for additive costs:

$$f(y) = \min_{d \in D} \{\pi(y',d) + f(y') \mid T(y',d) = y\}, \quad y \in (\Omega - y_0) \tag{1}$$

with the boundary condition

$$f(y_0) = k_0 \tag{2}$$

in which, $\Omega$ is the finite nonempty state space; $y_0 \in \Omega$ is the initial state; $d \in D$ is a decision, where D is the finite nonempty set of decisions; $T: \Omega \times D \to \Omega$ is the transition mapping, where $T(y',d)$ is the state that is reached when decision d is applied at state $y'$; $\pi: \Omega \times D \to \mathbb{R}$ is the cost function, where $\pi(y',d)$ is the incremental cost that is incurred when decision d is applied at state $y'$; $k_0 \in \mathbb{R}$ is the initial cost incurred in the initial state $y_0$; and in (2) we make the convenient but unnecessary assumption that return to the initial state is not possible. The functional equation (1) is simply a mathematical transliteration of Bellman's principle of optimality [3]. If we let $\Omega_F \subset \Omega$ be the set of final states, then the cost of an optimal policy (sequence of decisions) is $f^* = \min\{f(y') \mid y' \in \Omega_F\}$. Simply stated, an optimal policy is a sequence of decisions which, when applied at the initial state, reaches the set of final states at the minimum total cost.

The recursive solution of the functional equation to determine the cost of an optimal policy and the subsequent policy reconstruction process to determine the set of optimal policies is straightforward. However, the evaluation of $f(y)$ by (1) necessitates access to $f(y')$ in high-speed (magnetic core, thin film) computer storage for all states $\{y' \mid T(y',d) = y$ for some $d \in D\}$ and the policy reconstruction process necessitates access to the decisions at state $y'$ which result in $f(y)$ for <u>all</u> states in the state space $\Omega$ in low-speed (tape, disk) computer storage. It is common knowledge that in real problems excessive high-speed storage and computational requirements can present serious implementation problems, so much so that many problems cannot be solved even on the largest present day computers.

This paper presents a completely general approach to reducing both the high-speed and the low-speed computer storage requirements and, possibly, the computational requirements of dynamic programming algorithms. In order to accomplish this we invoke some of the elementary, but powerful, techniques of branch-and-bound. In particular we demonstrate that in the evaluation of $f(y)$ the costly procedure of storing $f(y')$ in high-speed memory for all states $\{y' | T(y',d) = y \text{ for some } d \in D\}$ may not always be necessary. If it can be demonstrated that there does not exist a sequence of decisions which when applied to state $y'$ will lead to an optimal policy, then it is not necessary to store $f(y')$ in high-speed memory (nor is it necessary to store the corresponding decision which led to state $y'$ in low-speed memory). Such states $y'$ can be identified by the use of relaxations and fathoming criteria when are commonly employed in branch-and-bound and other enumeration algorithms -- see [11,12,26,29,42] for example. Our approach has been inspired by the ingenious bounding schemes employed within dynamic programming algorithms in [37] and [43], by [11], by observations made by the authors in [32], and by the computational success of the resulting hybrid algorithm [28] -- see also [7,27,35,36].

The paper is organized in the following manner. The use of fathoming criteria and relaxations within dynamic programming algorithms is developed in § 1 for additive cost functions. The versatility of the result is manifested via application to a number of classes of problems in § 2. A numerical example of the traveling-salesman problem is presented and solved using the results of § 1. Computational experience on knapsack problems demonstrates the dramatic savings in both computer storage requirements and computation time which were effected by applying the results of § 1. In § 3 we show that our results extend immediately to general separable cost functions and discuss the alternative viewpoint of using dynamic programming within a branch-and-bound framework.

## 1. FATHOMING CRITERIA AND RELAXATIONS

Let $\mathcal{P}$ denote the original discrete optimization problem which we wish to solve. We will assume that the dynamic program $\mathcal{D}$ whose functional equation is (1) represents [22-24] the discrete optimization problem $\mathcal{P}$, so that we can employ (1) and (2) to solve $\mathcal{P}$. Let $\mathcal{U}$ be an upper bound on the objective function value of any optimal solution to the original discrete optimization problem $\mathcal{P}$. Then, since $\mathcal{D}$ is a representation of $\mathcal{P}$, it follows that $\mathcal{U}$ is also an upper bound on f*, i.e.,

$$\mathcal{U} \geq f^*. \tag{3}$$

Prior to discussing lower bounds and our results, it will be useful to define the policy space and some of its subsets and to define the transition mapping and cost function on the policy space. Let $\Delta$ denote the set of all policies and subpolicies (sequences of decisions) obtained by concatenating individual decisions in D. Let $\delta \in \Delta$ denote a subpolicy or policy, i.e. $\delta = (\delta(1), \delta(2), \ldots, \delta(n))$ where $\delta(i) \in D$, $i = 1,2,\ldots, n$. Extend the domains of both the transition mapping and the cost function from $\Omega \times D$ to $\Omega \times \Delta$ inductively and for any $\delta = \delta_1\delta_2$ define

$$T(y',\delta)=T(T(y',\delta_1), \delta_2), \text{ and}$$

$$\pi(y',\delta) = \pi(y',\delta_1) + \pi(T(y',\delta_1),\delta_2).$$

For any state $y' \in \Omega$ let $\Delta(y')$ denote the set of feasible subpolicies (policies if $y' \in \Omega_F$) which when applied to $y_0$ result in state $y'$, i.e., $\Delta(y') = \{\delta \in \Delta \mid T(y_0,\delta)= y'\}$, let $\Delta^*(y') \subseteq \Delta(y')$ denote the set of optimal subpolicies (policies if $y' \in \Omega_F$) for state $y'$, i.e., $\Delta^*(y') = \{\delta \in \Delta(y') \mid f(y') = k_0 + \pi(y_0,\delta)\}$, and let $\chi(y')$ denote the completion set of feasible subpolicies which when applied at state $y'$ result in a final state, i.e., $\chi(y')=\{\delta\in\Delta \mid T(y',\delta) \in \Omega_F\}$. The set of feasible policies for $\mathcal{D}$ is $\Delta(\Omega_F) = \bigcup_{y' \in \Omega_F} \Delta(y')$ and the set of optimal policies for $\mathcal{D}$ is $\Delta^* = \{\delta \in \Delta(\Omega_F) \mid f^* = k_0 + \pi(y_0,\delta)\}$.

For each state $y' \in \Omega$ we define a lower bound mapping $\ell: \Omega \to \mathbb{R}$ with the property that

$$\ell(y') \leq \pi(y',\delta) \qquad \forall \delta \in \chi(y'). \tag{4}$$

Then it is easily established that

PROPOSITION 1.1. If

$$f(y') + \ell(y') > \mathcal{U}, \tag{5}$$

then $\delta'\delta \notin \Delta^*$ for any $\delta' \in \Delta^*(y')$ and all $\delta \in \chi(y')$.

Proof. Substitute (3) and (4) into (5).

Simply stated, if the sum of the minimum cost of reaching state $y'$ from the initial state plus a lower bound on the cost of reaching any final state from state $y'$ exceeds an upper bound on the cost of an optimal policy, then no feasible completion of any optimal sub-policy for state $y'$ could be an optimal policy. Any state $y' \in \Omega$ which satisfies (5) is said to be fathomed [15,16]. The computer storage requirements are reduced since if $y'$ has been fathomed then $f(y')$ does not have to be placed in high-speed memory and the corresponding set of last decisions for $\delta' \in \Delta^*(y')$ does not have to be placed in low-speed memory. Furthermore, the computational requirements are also reduced since in the evaluation of $f(y)$ with the functional equation (1) it is not necessary to consider $f(y')$ for any fathomed state in $\{y' | T(y',d) = y \text{ for some } d \in D\}$.

An obvious upper bound $\mathcal{U}$ is the objective function value of any feasible solution to the original discrete optimization problem $\mathcal{P}$ or the value $(k_0 + \pi(y_0,\hat{\delta}))$ for any feasible policy $\hat{\delta} \in \Delta(\Omega_F)$. If the upper bound $\mathcal{U} = \hat{\mathcal{U}}$ is determined in this manner and we limit ourselves to identifying an optimal policy (instead of the entire set of optimal policies), then the following slightly stronger fathoming criterion can be employed.

PROPOSITION 1.2. *If*

$$f(y') + \ell(y') \geq \hat{\mathcal{U}}, \tag{6}$$

*then state* $y'$ *can be fathomed.*

Notice that the use of (6) allows for the possibility of verifying the optimality of the feasible policy $\hat{\delta}$ (sometimes referred to as the *incumbent* [16]) prior to the completion of the recursion, as will be demonstrated in the traveling-salesman example solved in the following section. This is a consequence of the following corollary to Proposition 1.2.

COROLLARY 1.1. *Suppose state* $y' \in \Omega$ *satisfies* (6) *and* $\delta' \in \Delta^*(y')$. *Then, all successive states* $y'' \in \Omega$ *such that* $\Delta(y'') = \{\delta'\delta'' | T(y',\delta'') = y''\}$ *can also be fathomed.*

Thus, if $\hat{\mathcal{U}} = \{k_0 + \pi(y_0,\hat{\delta})\}$ for some feasible policy $\hat{\delta}$ and we have the situation that there does not exist any state $y = T(y',d)$ for which $y' \notin \Omega_F$ has not been fathomed, then $\hat{\delta}$ is an optimal policy and we can stop the recursion at this point.

The lower bounds $\ell(y')$ can be obtained by solving a relaxed [15,16] version of the residual problem whose initial state is $y'$. If the cost function $\pi$ is nonnegative then an immediate lower bound for any state $y' \in \Omega$ is 0 corresponding to a total relaxation. However, much better lower bounds can usually be obtained by judicious choice of the relaxation as will be demonstrated in the following section.

Finally, we note that our approach includes conventional dynamic programming as a special case in which $\mathcal{U}$ is some sufficiently large real number and a total relaxation is employed.

## 2. EXAMPLES

In this section our results are illustrated and explicated vis-á-vis application to two specific problems. The traveling-salesman application is primarily of an illustrative nature, whereas the application to the nonlinear knapsack problem clearly demonstrates the computational power of our results.

### The Traveling-Salesman Problem

Let $\mathcal{G}_N = (V,E)$ be the directed graph whose vertex set is $V = \{1,2,\ldots, N\}$. Each arc $(i,j) \in E$ is assigned a nonnegative real weight $c_{ij}$. A tour (or Hamiltonian cycle) $t \in \mathcal{T}$ is a cycle which passes through each vertex exactly once. The tour $t = (1, i_2, \ldots, i_N, 1) \in \mathcal{T}$, where $(i_2, \ldots, i_N)$ is a permutation of the integers $(2,\ldots, N)$, can be given the ordered pair representation $t' = \big((1,i_2),(i_2,i_3),\ldots, (i_{N-1},i_N),(i_N,1)\big) \in \mathcal{T}'$. The traveling-salesman problem is to find a tour of minimum weight, i.e., find $t \in \mathcal{T}$ so as to

$$\min_{t' \in \mathcal{T}'} \sum_{(i,j) \in t'} c_{ij}. \tag{7}$$

If we think of the vertices as cities and the arcs as the distances between cities, the problem (7) is to start at some home city, visit all the N-1 other cities exactly once and then return home in the minimum total distance.

Consider the following dynamic programming algorithm [18] for problem (7). Let $f(S,j)$ denote the weight of a minimum weight path (subgraph) which starts at (contains) vertex 1, visits (contains) all vertices in $S \subseteq \{1,2,3,\ldots, N\}$, and terminates at vertex $j \in S$. Then, for all $S \neq \phi$, we have the following functional equations of a dynamic programming algorithm for the traveling-salesman problem

$$f(S,j) = \min_{i \in (S-j)} \{c_{ij} + f(S-j,i)\}, \tag{8}$$

with the boundary condition

$$f(\phi,-) = 0. \tag{9}$$

Notice that (8) and (9) are equivalent to (1) and (2) with $y_0 = (\phi,-)$, $y = (S,j) \in \Omega = \{[(S,j) | j \in S \subseteq \{2,3,\ldots, N\}] \cup (\{1,2,\ldots, N\},1)\}$, $y' = ((S-j), i)$, $d = j$, $T(y',d) = T((S-j,i),j) = (S,j)$, $\{(y',d) | T(y',d) = y\} = \{((S-j,i),j) | i \in (S-j)\}$, $\pi(y',d) = \pi((S-j,i),j) = c_{ij}$ and $k_0 = 0$.

The major roadblocks to solving even moderate size ($N \geq 20$) traveling-salesman problems by the dynamic programming algorithm, (8) and (9), are the excessive high-speed computer storage and computational requirements. The storage bottleneck occurs approximately halfway through the recursion when $n+1 = |S| = [N/2]$, since the evaluation of $f(S,j)$ by (8) requires high-speed access to $n\binom{N-1}{n}$ values of $f(S-j,i)$. Furthermore, even though the search over permutations in (7) has essentially been reduced to a recursive search over combinations in (8) and (9), the computational effort is still on the order of $N2^{N-1}$ (the exact number being $\sum_{n=2}^{N-1} n(n-1)\binom{N-1}{n} + (N-1) = (N-1)(1 + (N-2)(1 + (N-2)2^{N-3}))$. For $N = 20$, this amounts to a high-speed storage requirement of 92,378 locations and a computational requirement which involves performing over 1 million functional equation evaluations. However, we can reduce these requirements by fathoming certain states.

An upper bound on (7) is easily obtained as the weight of any tour $t \in \mathcal{T}$. For example, we could take the minimum of the weights of i) the tour (1,2,..., N-1, N,1) and ii) the N nearest-neighbor tours constructed by starting from each of the N cites. Alternatively, we

could take the weight of a tour $\hat{t}$ generated by some efficient heuristic as our upper bound $\hat{\mathcal{U}}$.

The lower bound $\ell(S,j)$ on the weight of a path $j \rightarrow 1$ which includes all vertices $k \notin S$ is also easily obtained. Since the assignment problem is a relaxation of (7) we can set $\ell(S,j)$ equal to the value of the minimum weight assignment to an assignment problem whose weight (cost) matrix $\hat{C}$ is obtained from C by deleting all rows $i \in \{1 \cup (S-j)\}$ and columns $k \in S$. A relaxation of the traveling-salesman problem which can be solved by a "greedy" algorithm is the minimum weight 1 - tree problem which is a minimum (weight) spanning tree on the vertex set $\{2,\ldots, N\}$ with the two edges of lowest weight adjoined at vertex 1. Other relaxations have been discussed in [5,7,19,20,42].

Suppose that we have calculated the bounds $\hat{\mathcal{U}}$ and $\ell(S,j)$. Then Proposition 1.2 can be stated for the traveling-salesman problem as follows:

PROPOSITION 2.1. T-S Fathoming Criterion: If

$$f(S,j) + \ell(S,j) \geq \hat{\mathcal{U}} \tag{10}$$

then any tour $t \in \mathcal{T}$ which contains a path between vertex 1 and vertex j that connect all vertices in S-j cannot have a lower weight than tour $\hat{t} \in \mathcal{T}$ which has weight $\hat{\mathcal{U}} = \sum_{(i,j)\in\hat{t}'} c_{ij}$. Hence, state (S,j) can be fathomed.

As a consequence of Corollary 1.1 we also have

COROLLARY 2.1. If for some $|S| = n < N$ all states (S,j) for which $|S| = n$ and $j \in S$ are fathomed by (10), then $\hat{t}$ is an optimal tour.

Corollary 2.1 allows for the possibility of verifying the optimality of an initial tour $\hat{t}$ prior to the completion of the recursion.

The use of Proposition 2.1 and Corollary 2.1 is demonstrated on the following simple numerical example.

Numerical Example: Wagner [42, p.472]

Consider the directed graph $\mathcal{G}_5$ on the vertex set $V = \{1,2,\ldots, 5\}$ whose weight matrix C is

$$C = \begin{bmatrix} L & 10 & 25 & 25 & 10 \\ 1 & L & 10 & 15 & 2 \\ 8 & 9 & L & 20 & 10 \\ 14 & 10 & 24 & L & 15 \\ 10 & 8 & 25 & 27 & L \end{bmatrix}$$

where $L \in \mathbb{R}_+$ is sufficiently large.

The calculations involved in the solution of the functional equation (8) and (9), are summarized in Table I. The weight of the optimal tour $\{1,5,2,3,4,1\}$ is 62.

TABLE I. Summary of Calculations for the D.P. Solution of the Traveling-Salesman Example Problem

| $S = \phi$ | | | $\lvert S\rvert = 1$ | | | $\lvert S\rvert = 2$ | | | $\lvert S\rvert = 3$ | | | $\lvert S\rvert = 4$ | | | $\lvert S\rvert = 5$ | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| (S,i) | f(S,i) | i* | (S,i) | f(S,i) | i* | (S,i) | f(S,i) | i* | (S,i) | f(S,i) | i* | (S,i) | f(S,i) | i* | (S,i) | f(S,i) | i* |
| (φ,-) | 0 | - | ([2],2) | 10 | 1 | ([2,3],2) | 34 | 3 | ([2,3,4],2) | 55 | 4 | ([2,3,4,5],2) | 65 | 4 | ([1,2,3,4,5],1) | 62 | 4 |
| | | | ([3],3) | 25 | 1 | ([2,3],3) | 20 | 2 | ([2,3,4],3) | 45 | 2 | ([2,3,4,5],3) | 57 | 2 or 4 | | | |
| | | | ([4],4) | 25 | 1 | ([2,4],2) | 35 | 4 | ([2,3,4],4) | 40 | 3 | ([2,3,4,5],4) | 48 | 3 | | | |
| | | | ([5],5) | 10 | 1 | ([2,4],4) | 25 | 2 | ([2,3,5],2) | 43 | 5 | ([2,3,4,5],5) | 55 | 3 or 4 | | | |
| | | | | | | ([2,5],2 | 18 | 5 | ([2,3,5],3) | 28 | 2 | | | | | | |
| | | | | | | ([2,5],5) | 12 | 2 | ([2,3,5],5) | 30 | 3 | | | | | | |
| | | | | | | ([3,4],3) | 49 | 4 | ([2,4,5],2) | 47 | 4 | | | | | | |
| | | | | | | ([3,4],4) | 45 | 3 | ([2,4,5],4) | 33 | 2 | | | | | | |
| | | | | | | ([3,5],3) | 35 | 5 | ([2,4,5],5) | 37 | 2 | | | | | | |
| | | | | | | ([3,5],5) | 35 | 3 | ([3,4,5],3) | 61 | 4 | | | | | | |
| | | | | | | ([4,5],4) | 37 | 5 | ([3,4,5],4) | 55 | 3 | | | | | | |
| | | | | | | ([4,5],5) | 40 | 4 | ([3,4,5],5) | 59 | 3 | | | | | | |

We next solve this example problem with the hybrid DP/branch-and-bound algorithm, i.e., with (8) and (9) incorporating the fathoming criterion (10). The upper bound $\hat{U}$ is 62: the minimum of the weight (65) of the tour (1,2,3,4,5,1) and the weight (62) of a nearest-neighbor tour $\hat{t}$ = (1,5,2,3,4,1). The lower bounds $\ell(S,j)$ will be calculated by solving the assignment problem relaxation.

For $|S| = 1$, we have the following calculations:

| (S,j) | f(S,j) | i* | $\ell(S,j)$ |
|---|---|---|---|
| ({2},2) | 10 | 1 | 55 |
| ({3},3) | 25 | 1 | 42 |
| ({4},4) | 25 | 1 | 40 |
| ({5},5) | 10 | 1 | 50 |

Notice that we can fathom states {({2},2), ({3},3), ({4},4)} immediately by (10), since we have

$$f(S,j) + \ell(S,j) \geq \hat{U}.$$

Therefore, the maximum high-speed storage for $|S| = 1$ is 1 location as opposed to 4 locations in the conventional DP approach -- only information on state ({5},5) is relevant to the calculation of f(S,j) for $|S| = 2$.

Furthermore, by Corollary 1.1 we can fathom states {({2,3},2), ({2,3},3), ({2,4},2), ({2,4},4), ({2,5},5), ({3,4},3), ({3,4},4), ({3,5},5), ({4,5},5)} even before evaluating either f(S,j) or $\ell(S,j)$ since they clearly could not lead to tours which have lower weights than $\hat{t}$. Therefore, for $|S| = 2$ only 3 of the 12 possible states remain. The calculations for these states are presented below:

| (S,j) | f(S,j) | i* | ℓ(S,j) |
|---|---|---|---|
| ({2,5},2) | 18 | 5 | 44 |
| ({3,5},3) | 35 | 5 | 31 |
| ({4,5},4) | 37 | 5 | 28 |

We can fathom all of these states by (10) indicating that $\hat{t}$ is indeed an optimal tour. By Corollary 2.1 no additional computations are required and the recursion can be halted. The drastic reductions in both storage and computational requirements which were effected by employing the fathoming criterion within the DP algorithm are summarized in Table II.

Reference to Table II reveals that the high-speed storage requirement of the hybrid algorithm is 1/12 that of DP and that the low-speed storage requirement of the hybrid algorithm is 1/33 that of DP in the solution of this example problem! This was partly attributable to the very good (in fact, optimal) initial tour $\hat{t}$. However, the reader should verify that even if the crudest upper bound $\hat{\hat{u}} = 65$ corresponding to the tour $\hat{\hat{t}} = (1,2,3,4,5,1)$ is used that the recursion could be halted at the end of the evaluations for $|S| = 3$ since only state ({2,3,5},3) is unfathomed. (Therefore the only possible tour goes to 4, then 1 and this is optimal.) The storage and computational requirements of the hybrid DP/branch-and-bound algorithm with the crude bound $\hat{\hat{u}} = 65$ are summarized below

| \|S\| | No. of states for which f(S,j) is evaluated | Max. no. of unfathomed states | Evaluation of f(S,j) No. of Additions | Evaluation of f(S,j) No. of Comparisons | No. of bounds ℓ(S,j) calculated |
|---|---|---|---|---|---|
| 1 | 4 | 2 | 0 | 0 | 4 |
| 2 | 6 | 1 | 6 | 0 | 6 |
| 3 | 2 | 1 | 2 | 0 | 2 |
| 4 | - | - | - | - | - |
| 5 | - | - | - | - | - |
| Σ = | 12 | 4 | 8 | 0 | 12 |

maximum high-speed storage requirement = 2

total low-speed storage requirement = 3

total number of states fathomed = 29

TABLE II. Summary of Reduction in Both Storage and Computational Requirements Effected by Employing Fathoming Criteria in the DP Solution of the Traveling-Salesman Example Problem.

Conventional DP Solution

| \|S\| | No. of States for which f(S,j) is evaluated | Evaluation of f(S,j) No. of Additions | No. of Comparisons |
|---|---|---|---|
| 1 | 4 | 0 | 0 |
| 2 | 12 | 12 | 0 |
| 3 | 12 | 24 | 12 |
| 4 | 4 | 12 | 4 |
| 5 | 1 | 4 | 1 |
| $\Sigma$ = | 33 | 48 | 17 |

max high-speed storage requirement = 12

total low-speed storage requirement = 33

Solution by the Hybrid DP /Branch-and-Bound Algorithm ($\hat{U}$ = 62)

| \|S\| | No. of States for which f(S,j) is evaluated | Max no. of unfathomed states | Evaluation of f(S,j) No. of Additions | No. of Comparisons | No. of bounds $\ell$(S,j) calculated |
|---|---|---|---|---|---|
| 1 | 4 | 1 | 0 | 0 | 4 |
| 2 | 3 | 0 | 3 | 0 | 3 |
| 3 | - | - | - | - | - |
| 4 | - | - | - | - | - |
| 5 | - | - | - | - | - |
| $\Sigma$ = | 7 | 1 | 3 | 0 | 7 |

max high-speed storage requirement = 1

total low-speed storage requirement = 1

total number of states fathomed = 32

This traveling-salesman example is primarily of an illustrative nature and the hybrid approach may not be competititve with other relatively efficient algorithms [19,20] for the traveling-salesman problem. However, we are currently experimenting with a FORTRAN IV code of the hybrid algorithm (written by Richard King of Northwestern) and have solved the symmetric 10-city problem of [7] in 0.696 CPU seconds on a CDC 6400 with the code, fathoming 1733 of the 2305 possible states in the process.

## The Nonlinear Knapsack Problem [32]

The nonlinear knapsack problem (NKP) can be stated as follows: find $x \in \mathbb{R}_+^N$ so as to

$$\text{maximize} \sum_{j=1}^{N} r_j(x_j) \tag{11}$$

$$\text{subject to} \sum_{j=1}^{N} g_{ij}(x_j) \le b_i \qquad i = 1,2,\ldots, M$$

$$x_j \in S_j \qquad j = 1,2,\ldots, N$$

where $(\forall j)\ S_j = \{0,1,\ldots, K_j\}$ and $r_j: S_j \to \mathbb{R}_+$ is nondecreasing with $r_j(0) = 0$, $(\forall ij)\ g_{ij}: S_j \to \mathbb{R}_+$ with $g_{ij}(0) = 0$, and $b = (b_1, b_2, \ldots, b_M) \ge 0$. The (NKP) includes both the classical (one-dimensional) knapsack problem [39] and the multidimensional 0/1 knapsack problem [43] as special cases. The (NKP) also includes a number of other varients [12] of the knapsack problem and the "optimal distribution of effort problem" [25,42] as well as various discrete nonlinear resource allocation problems [9,13] graph-theoretic problems [14,41] and nonlinear integer programming problems [15,40], as special cases.

Consider the following (imbedded state space) dynamic programming algorithm, M&MDP [32], for the solution of (11). Let $f(n,\beta)$ denote the maximum objective function value of an undominated feasible solution to (11) in which at most all of the first n variables $(x_1,x_2,\ldots, x_n)$ are positive and whose resource consumption vector does not exceed $\beta = (\beta_1,\beta_2,\ldots, \beta_M)$, i.e., $(\forall i)\ \beta_i \geq \sum_{j=1}^{n} g_{ij}(x_j)$. For $n = 1,2,\ldots, N$, the feasible solution $x = (x_1,x_2,\ldots, x_n)$ is said to be <u>dominated</u> by the feasible solution $\hat{x} = (\hat{x}_1,\hat{x}_2,\ldots, \hat{x}_n)$ if we have both $\sum_{j=1}^{n} r_j(x_j) \leq \sum_{j=1}^{n} r_j(\hat{x}_j)$ and $\sum_{j=1}^{n} g_{ij}(x_j) \geq \sum_{j=1}^{n} g_{ij}(\hat{x}_j)$ with strict inequality holding in at least one of the (M+1) inequalities. For $1 \leq n \leq N$, let $F_n$ be the (domain) set of resource consumption vectors, $\beta$, of all undominated feasible solutions $(x_1,x_2,\ldots, x_n)$ to the following subproblem

$$\max \sum_{j=1}^{n} r_j(x_j) \tag{12}$$

$$\text{subject to} \sum_{j=1}^{n} g_{ij}(x_j) \leq b_i \qquad i = 1,2,\ldots, M$$

$$x_j \in S_j \qquad j = 1,2,\ldots, n$$

Also, for $0 \leq n \leq N$, let $V_n$ be the (domain) set of resource consumption vectors $g_n(k) = (g_{1n}(k), g_{2n}(k),\ldots, g_{Mn}(k))$, of all undominated feasible values of $x_n = k \in S_n$. We can demonstrate [32] that for $1 \leq n \leq N$, $F_n \subseteq (V_n \circledast F_{n-1})$ with $F_0 = \phi$, where $(V_n \circledast F_{n-1})$ denotes the set obtained by forming all sums of one element of $V_n$ with one element of $F_{n-1}$.

Therefore, for $1 \leq n \leq N$ we can recursively generate all feasible candidates for $F_n$ from $F_{n-1}$ and $V_n$ with the following functional equation of M&MDP:

$$f(n,\beta) = \{r_n(k) + f(n-1,\beta - g_n(k)) \mid g_n(k) \in V_n\ ,\ (\beta - g_n(k)) \in F_{n-1}\ ,\ \beta \leq b\} \tag{13}$$

with the boundary condition

$$f(0,0) = 0. \tag{14}$$

If the $f(n,\beta)$ corresponding to dominated solutions are eliminated at the end of (or during) each stage n, then (13) and (14) will yield the set of objective function values $\{f(N,\beta) \mid \beta \in F_N\}$ of the complete family of undominated feasible solutions to the (NKP). Notice that the functional equations, (13) and (14) together with the dominance elimination, are equivalent to (1) and (2) with $y_0 = (0,0)$, $y = (n,\beta) \in \Omega = \{(n,\beta) \mid \beta \in F_n,\ n = 0,1,\ldots, N\}$, $y' = (n\text{-}1,\ \beta - g_n(k))$, $d = k$, $T(y',d) = T((n\text{-}1,\ \beta - g_n(k)),k) = (n,\beta) \in \Omega$, $\{(y',d) \mid T(y',d) = y\} = \{(n\text{-}1,\beta - g_n(k)) \mid g_n(k) \in V_n,\ (\beta - g_n(k)) \in F_{n-1},\ \beta \leq b\}$, $\pi(y',d) = r_n(k)$, and $k_0 = 0$.

M&MDP is basically a one-dimensional recursive search (with dominance elimination) over the sequence of sets $F_0, F_1, \ldots, F_N$ which are all imbedded in the M-dimensional space $B = \{\beta \mid (\forall i)\ \beta_i \in [0,b_i]\}$. As a result of this "imbedded state space" approach [31], M&MDP is not overcome by the "curse of dimensionality" which inhibits the utility of conventional dynamic programming algorithms on higher dimensional problems. However, both the high-speed storage requirements and the computational requirements are directly related to the length $|F_n|$ of the list $F_n$ of undominated feasible solutions at stage n. Specifically, the high-speed storage requirement is on the order of $(2M + 6)\bar{n}$ where $\bar{n} = \max_n\{|F_n|\}$ = maximum list length and, although a very efficient dominance elimination scheme (which eliminates sorting through the use of (M+1) threaded lists) is employed in M&MDP, the computational time tends to increase exponentially with $\bar{n}$ -- in fact, problems in which $\bar{n} \geq 10{,}000$ might consume hours of computer time. Fortunately, we can employ the fathoming and relaxation results of § 1 (with minima replaced by maxima, upper

bounds replaced by lower bounds, and vice versa) to great advantage in reducing the list lengths.

Let $\mathcal{L}$ be a lower bound on the value of an optimal solution $x \in X^*$ to the (NKP). For any $(n,\beta) \in \Omega$, $f(n,\beta)$ is the value of an optimal solution $\delta' = x' = (x'_1, x'_2, \ldots, x'_n) \in \Delta^*(n,\beta)$ to subproblem (12) for $b = \beta$. Consider the following (residual) subproblem for any $(n,\beta) \in \Omega$:

Find $(x_{n+1}, x_{n+2}, \ldots, x_N) \in \mathbb{R}_+^{N-n}$ so as to

$$\max \sum_{j=n+1}^{N} r_j(x_j) \tag{15}$$

subject to

$$\sum_{j=n+1}^{N} g_{ij}(x_j) \leq b_i - \beta_i \qquad i = 1,2,\ldots, M$$

$$x_j \in S_j \qquad j = n+1, n+2, \ldots, N$$

Let $\mathcal{X}(n,\beta)$ denote the set of all feasible solutions to subproblem (15). Let $u(n,\beta)$ be an upper bound on the value of an optimal solution to the residual subproblem (15). Then, Proposition 1.1 can be stated for the (NKP) as follows:

PROPOSITION 2.2. (NKP) Fathoming Criterion: If for any $(n,\beta) \in \Omega$, we have

$$f(n,\beta) + u(n,\beta) < \mathcal{L} \tag{16}$$

then $\nexists\ \hat{x} \in \mathcal{X}(n,\beta)$ such that $x'\hat{x} \in X^*$.

Any $\beta \in F_n$ for which (16) holds (for the corresponding $(n,\beta) \in \Omega$)

can be eliminated from $F_n$, thereby reducing the lengths of the lists $F_n, F_{n+1}, \ldots, F_N$. Since $\beta \in F_n$ may give rise to $\prod_{j=n+1}^{k} (K_j + 1)$ members in list $F_k (n+1 \leq k \leq N)$ the effect of fathoming can be significant.

The choice of bounds and relaxations is dependent on the form of both the return functions and the constraints and is discussed in detail in [28]. The objective function value of any feasible solution to the (NKP) provides a lower bound $\mathcal{L}$. One easily computed upper bound $u(n,\beta)$ for any state $(n,\beta) \in \Omega$ is $\sum_{j=n+1}^{N} r_j(K_j)$; another is $\min_{1 \leq i \leq M} \{\rho_{in}(b_i - \beta_i)\}$ where

$$\rho_{in} = \max_{n+1 \leq j \leq N} \left( \max_{1 \leq k \leq K_j} \{r_j(k)/g_{ij}(k)\} \right) \tag{17}$$

We note that for linear problems, as a result of our solution tree structure, linear programming relaxations can be used to particular advantage. That is, in our procedure there is no backtracking and the same LP relaxation of (15) applies to all $\beta \in F_n$ with only the $(b-\beta)$ vector changing value. For fixed n the dual objective function value of any dual feasible solution of this relaxation is an upper bound for all $\beta \in F_n$, allowing dual solutions to be shared.

The hybrid (M&MDP with fathoming) algorithm was coded in FORTRAN IV and applied to several of the "hard" linear problems which were solved with M&MDP [32]. The computational experience with both the hybrid algorithm and the straight M&MDP are presented in Table III. The times presented are the execution (CPU) times in seconds of the codes on M.I.T.'s IBM 370-165.

TABLE III. Computational Comparison of the Hybrid Algorithm vs. M&MDP

| Problem No. | Problem Size N | M | K | M&MDP $\max_n \lvert F_n \rvert$ | M&MDP execution time (sec.) | Hybrid Algorithm $\max_n \lvert F_n \rvert$ | Hybrid Algorithm execution time (sec.) |
|---|---|---|---|---|---|---|---|
| 4 | 15 | 10 | 1 | 391 | 6.36 | 16 | 0.29 |
| 5 | 20 | 10 | 1 | 526 | 6.26 | 22 | 0.31 |
| 6 | 28 | 10 | 1 | 814 | 24.18 | 62 | 0.81 |
| 7 | 39 | 5 | 1 | 1043 | 11.27 | 51 | 0.92 |
| 8 | 50 | 5 | 1 | 2053 | 150.00 | 49 | 1.33 |
| 12 | 34 | 5 | 1 | 540 | 8.28 | 29 | 0.38 |
| 13 | 50 | 5 | 1 | 571 | 12.48 | 12 | 0.42 |

In all problems of Table III, the decision variables were sequenced in nonincreasing order of $r_j(1)$ for input to both algorithms. The simple $\min_{1 \le i \le M} \{\rho_{in}(b_i - \beta_i)\}$ upper bounds were used in the hybrid algorithm. When Problem 8 was solved with the LP relaxations employed at every state in the hybrid algorithm the $\max_n \lvert F_n \rvert = 3$ and the execution time was 4.99 seconds -- Since 0 is a member of the $F_n$ list the savings were on the order of $10^3$ (2052 vs. 2) and the computational savings were on the order of $10^1$ (150.00 seconds vs. 4.99 seconds) with the hybrid algorithm.

Reference to Table III reveals the dramatic reduction in both computer storage requirements (which are a function of $\max_n \lvert F_n \rvert$) and computation times which were affected by incorporating the fathoming criteria and relaxations within M&MDP. In fact, on the basis of the additional computational experience reported in [28], the hybrid algorithm appears to be competitive with even the best available linear 0/1 integer programming algorithms for this special case of the (NKP) and the hybrid algorithm has the ability to solve a much broader class of problems.

## 3. EXTENSIONS AND DISCUSSION

Our results extend immediately to dynamic programs with separable costs simply by replacing addition + with the composition operator o [8,29,34] whenever it occurs. The composition operator o represents any cummutative isotonic binary operator, some common examples of which are addition (+), multiplication (x), and the infix operators [4] $\vee$ and $\wedge$, where for $a \neq b \in \mathbb{R}$, $a \vee b = \max\{a,b\}$ and $a \wedge b = \min\{a,b\}$. Our results can also be extended to dynamic programs with general cost functions as in KARP and HELD [24] and IBARAKI [22,23] where discrete dynamic programs are viewed as finite automata [6,38] with a certain cost structure superimposed upon them. That treatment allows for the analysis of the nontrivial representation problem [24] and requires a somewhat different development which can be found in [33]. In both the separable and the general cost function extensions we assume that the cost functions are monotone [8,24,29,34]. This is sufficient to insure that the resulting dynamic programming algorithm solves the dynamic program $\mathcal{D}$. Furthermore, in the case of multiplicative returns ("o" = "x") we also require that the cost functions be nonnegative.

The principle ideas of this paper may be viewed as dynamic programming plus bounds (as we have done herein) or, alternatively, as branch-and-bound plus dominance. In the latter case we could say that node A in a branch-and-bound tree dominates node B if it can be shown that no feasible solution obtained as a descendent of B can be better than any optimal descendent of A. Such a strategy has proven useful on the solution of a number of problems [1,2,10,21,30]. Notice that this alternative could also be broadly interpreted as using dynamic programming within a branch-and-bound framework since the dominance elimination is analogous to the initial fathoming which is employed in conventional dynamic programming.

## 4. CONCLUSION

It has been demonstrated that both the computer storage requirements and also the computational requirements of dynamic programming algorithms can be significantly reduced through the use of branch-and-bound methods. We envision that this hybrid approach may open the doorways to the solutions of many sizes and classes of problems which are currently computationally "unsolvable" by conventional dynamic programming algorithms.

## ACKNOWLEDGEMENTS

The authors wish to thank GEORGE NEMHAUSER and the two referees for their useful suggestions.

REFERENCES

1. J. H. AHRENS and G. FINKE, "Merging and Sorting Applied to the Zero-One Knapsack Problem," Opns. Res., forthcoming.

2. K. R. BAKER, Introduction to Sequencing and Scheduling, John Wiley and Sons, New York, 1974.

3. R. E. BELLMAN, Dynamic Programming, Princeton University Press, Princeton, N.J., 1957.

4. _____ and L. A. ZADEH, "Decision-Making in a Fuzzy Environment," Management Sci., 17, B141-B164 (1970).

5. M. BELLMORE and G. L. NEMHAUSER, "The Traveling-Salesman Problem: A Survey," Opns. Res., 16, 538-558 (1968).

6. T. L. BOOTH, Sequential Machines and Automata Theory, John Wiley and Sons, New York, N.Y., 1967.

7. N. CHRISTOFIDES, "Bounds for the Traveling-Salesman Problem," Opns. Res. 20, 1044-1056 (1972).

8. E. V. DENARDO and L. G. MITTEN, "Elements of Sequential Decision Processes," J. Industrial Engineering 18, 106-112 (1967).

9. V. DHARMADHIKARI, "Discrete Dynamic Programming and the Nonlinear Resource Allocation Problem," Technical Report CP-74009, Department of Computer Science and Operations Research, Southern Methodist University, Dallas, Texas, March, 1974.

10. S. E. ELMAGHRABY, "The One-Machine Sequencing Problem with Delay Costs," J. Industrial Engineering, 19, 105-108 (1968).

11. A. O. ESOGBUE, Personal communication, 1970.

12. B. FAALAND, "Solution of the Value-Independent Knapsack Problem by Partitioning," Opns. Res. 21, 333-337 (1973).

13. B. FOX, "Discrete Optimization Via Marginal Analysis," Management Sci., 13, 210-216 (1966).

14. H. FRANK, L. T. FRISCH, R. VAN SLYKE, and W. S. CHOU, "Optimal Design of Centralized Computer Networks," Networks 1, 43-57 (1971).

15. R. S. GARFINKEL and G. L. NEMHAUSER, Integer Programming, Wiley-Interscience, New York, N.Y., 1972.

16. A. M. GEOFFRION and R. E. MARSTEN, "Integer Programming Algorithms: A Framework and State-of-the-Art-Survey," Management Sci. 18, 465-491 (1972).

17. G. GRAVES and A. WHINSTON, "A New Approach to Discrete Mathematical Programming," Management Sci. 15, 177-190 (1968).

18. M. HELD and R. M. KARP, "A Dynamic Programming Approach to Sequencing Problems," J. SIAM 10, 196-210 (1962).

19. _____ and _____, "The Traveling-Salesman Problem and Minimum Spanning Trees," Opns. Res. 18, 1138-1162 (1970).

20. _____ and _____, "The Traveling-Salesman Problem and Minimum Spanning Trees - Part II," Mathematical Programming 1, 6-25 (1971).

21. E. IGNALL and L. SCHRAGE, "Application of the Branch-and-Bound Technique to Some Flow-Shop Scheduling Problems," Opns. Res. 11, 400-412 (1965).

22. T. IBARAKI, "Finite State Representations of Discrete Optimization Problems," SIAM J. Computing 2, 193-210 (1973).

23. _____, "Solvable Classes of Discrete Dynamic Programming," J. Math. Anal. and Appl. 43, 642-693 (1973).

24. R. M. KARP and M. HELD, "Finite-State Processes and Dynamic Programming," SIAM J. Appl. Math. 15, 693-718 (1967).

25. W. KARUSH, "A General Algorithm for the Optimal Distribution of Effort," Management Sci. 9, 229-239 (1962).

26. E. L. LAWLER and D. E. WOOD, "Branch and Bound Methods: A Survey," Opns. Res. 14, 699-719 (1966).

27. J. B. MACQUEEN, "A Test for Suboptimal Actions in Markovian Decision Problems," Opns. Res. 15, 559-561 (1967).

28. R. E. MARSTEN and T. L. MORIN, "A Hybrid Approach to Discrete Mathematical Programming," Working Paper, Sloan School of Management, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1974.

29. L. G. MITTEN, "Composition Principles for Synthesis of Optimal Multistage Process," Opns. Res. 12, 610-619 (1964).

30. _____, "Branch-and-Bound Methods: General Formulation and Properties," Opns. Res. 18, 24-34 (1970).

31. T. L. MORIN and A. M. O. ESOGBUE, "The Imbedded State Space Approach to Reducing Dimensionality in Dynamic Programs of Higher Dimensions," J. Math. Anal. and Appl. 48, 801-810 (1974).

32. _____ and R. E. MARSTEN, "An Algorithm for Nonlinear Knapsack Problems," Technical Report No. 95, Operations Research Center, Massachusetts Institute of Technology, Cambridge, Massachusetts, May, 1974, to appear in Management Sci.

33. _____ and _____, "Branch-and-Bound Strategies for Dynamic Programming," Discussion Paper No. 106, The Center For Mathematical Studies in Economics and Management Science, Northwestern University, Evanston, Illinois, Sept. 1974.

34. G. L. NEMHAUSER, Introduction to Dynamic Programming, John Wiley and Sons, New York, N.Y., 1966.

35. _____, "A Generalized Permanent Label Setting Algorithm for the Shortest Path Between Specified Nodes," J. Math. Anal. and Appl. 38, 328-334 (1972).

36. E. L. PORTEUS, "Some Bounds for Discounted Sequential Decision Processes," Management Sci. 18, 7-11 (1971).

37. F. PROSCHAN and T. A. BRAY, "Optimal Redundancy Under Multiple Constraints," Opns. Res. 13, 800-814 (1965).

38. M. D. RABIN and D. SCOTT, "Finite Automata and their Decision Problems," IBM J. Res. and Development 3, 114-125 (1959).

39. H. M. SALKIN and C. A. DE KLUYVER, "The Knapsack Problem: A Survey," Technical Memorandum No. 281, Department of Operations Research, Case Western Reserve University, Cleveland, Ohio, December, 1972.

40. R. E. SCHWARTZ and C. L. DYM, "An Integer Maximization Problem," Opns. Res. 19, 548-550 (1971).

41. R. VAN SLYKE and A. KERSHENBAUM, "Network Analysis by Accumulation of Trees," Lecture Notes for the course, Mathematical Programming for Management Decision: Theory and Application, Massachusetts Institute of Technology, Cambridge, Massachusetts, July, 1973.

42. H. M. WAGNER, Principles of Operations Research, Prentice-Hall, Englewood Cliffs, N.J., 1969.

43. H. M. WEINGARTNER and D. N. NESS, "Methods for the Solution of the Multi-Dimensional 0/1 Knapsack Problem," Opns. Res. 15, 83-103 (1967).